HDL-TR-2192

March 1991

EMP Code Performance Comparisons for IBM RS 6000 Workstations, Models 320 and 530

by William T. Wyatt, Jr., and Christopher S. Kenyon

U.S. Army Laboratory Command
Harry Diamond Laboratories
Adelphi, MD 20783-1197

The findings in this report are not to be construed as an official Department of the Army position unless so designated by other authorized documents.

Citation of manufacturer's or trade names does not constitute an official endorsement or approval of the use thereof.

Destroy this report when it is no longer needed. Do not return it to the originator.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY *(Leave blank)*	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	March 1991	Interim, from 1 Aug to 1 Sept 1990

4. TITLE AND SUBTITLE
EMP Code Performance Comparisons for IBM RS 6000 Workstations, Models 320 and 530

5. FUNDING NUMBERS
PE: 62715H

6. AUTHOR(S)
William T. Wyatt, Jr., and Christopher S. Kenyon

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Harry Diamond Laboratories
2800 Powder Mill Road
Adelphi, MD 20783-1197

8. PERFORMING ORGANIZATION REPORT NUMBER
HDL-TR-2192

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
U.S. Army Laboratory Command
2800 Powder Mill Road
Adelphi, MD 20783-1145

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES
AMS code: 9700400.4201
HDL PR: E300E3

12a. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution unlimited.

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

Executing electromagnetic pulse (EMP) codes on personal computers and workstations, in addition to mainframe supercomputers, has been a goal of research in recent years. Until now, personal computers and workstations have offered insufficient resources to make this goal reachable. We present some results of limited comparisons made between IBM RISC System 6000 model 320 and 530 workstations, several 80286 and 80386 personal computers, and IBM 3090 and Cray X-MP mainframe supercomputers. We emphasized floating-point speed in execution of several small, medium, and large Fortran-based computer codes, including a number of EMP codes. We show some additional results for the Dhrystone benchmark. The comparisons clearly place the RS 6000 workstations close to the mainframe systems tested in speed and memory capacity. The RS 6000/530 system ran the Dhrystone benchmark three times faster than the Cray X-MP, and ran small scalar Fortran benchmarks as fast as the Cray X-MP. Some large codes, which were able to restrict frequent data references to the RS 6000 high-speed memory cache, ran about two times faster than other large codes. Cache-efficient medium-sized codes ran faster than other medium-sized codes, but not as much as a factor of two. The "xlf" Fortran compiler for the RS 6000 performed very well on severe tests of reliability and code optimization. We conclude that the RS 6000 models tested can run mainframe-class EMP codes close to the user and his interactive graphical interface.

14. SUBJECT TERMS	15. NUMBER OF PAGES
Benchmark, Fortran, Whetstone, Dhrystone, workstation, EMP, computer, software, IBM, RISC	22
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	17. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	UL

NSN 7540-01-280-5500

Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. Z39-18
298-102

Contents

Tables

1. Introduction

In recent years, some effort has been made to determine whether personal computers or workstations of similar size can carry out calculations of the electromagnetic pulse (EMP) environment produced by nuclear explosions. Generally, such calculations involve time-dependent finite-difference solutions in one, two, or three spatial dimensions. Supporting calculations are often done in the frequency domain and transformed to the time domain by fast Fourier transform (FFT) techniques. To date, only a few codes for EMP calculation have been successfully ported to such small systems because of the speed, memory, and disk storage limitations of personal computers and workstations.

This report presents some results of a series of limited comparisons made between an IBM RISC System 6000, model 320 (hereinafter called an RS 6000/320), an IBM RISC System 6000, model 530 (hereinafter called an RS 6000/530), and a selection of computers available to the authors, varying in speed from an IBM PC-AT personal computer to a Cray X-MP supercomputer. To enable us to make these comparisons, IBM made the RS systems available to us for several weeks. We made the comparisons to demonstrate where the RS systems fall in the spectrum of processors available at the Harry Diamond Laboratories (HDL), and to illustrate their potential application to EMP codes.

The RS series is a new workstation product from IBM based on a set of processor chips which are an outgrowth of the RT technology marketed in the 1980's. The RT technology was a first-generation reduced instruction set computer (RISC) design using concepts developed in the IBM 801 minicomputer. The new RS processor chips are second-generation RISC in concept, with a smaller instruction set than found in IBM mainframe processors, but with more powerful instructions than commonly found in previous RISC systems. The RS reduced instruction set implements a sophisticated architecture, and combines into single instructions several functions which would take two or more instructions in many other systems. The instruction lookahead capability is powerful enough to perform zero-cycle branching. Loop count instructions can be done in parallel with floating-point instructions. In combination these features allow computational loops with no loop overhead. In addition, all floating-point operations are implemented as special cases of a single, fast, multiply-and-add primitive, allowing up to two floating-point operations to be completed every clock period. Many other clever design enhancements have produced a system of extraordinary power.

We do not go into further details of the RS 6000 design, since these are well documented [1]. However, we do point out three important differences between the RS 6000 model 320 and model 530:

- 50-ns clock period in the 320 and 40-ns clock period in the 530;
- 32-kbyte data cache in the 320 and 64-kbyte data cache in the 530; and
- 64-bit-wide memory data bus in the 320 and 128-bit-wide memory data bus in the 530.

We address only the computational speed of the RS 6000/320 and RS 6000/530 in connection with floating-point problems and, briefly, with the Dhrystone [2] and Landmark [3] benchmarks. Although the systems have extensive graphics capability, graphics is not within our special expertise, and we made no attempt to compare graphics performance.

2. Compatibility With Other Systems

The RS 6000 uses AIX, the IBM implementation of Unix, as the native operating system, with full TCP/IP network support. Our access to the RS 6000 was through either the system console, often using a mouse-driven X-windows interface, or an Ethernet local area network via TCP/IP network protocol, supporting Telnet virtual terminal and FTP file transfer applications. No problems were encountered in the network interface. One of the HDL test personnel could occasionally "hang" the console X-windows interface, but was unable to derive which particular sequence of operations caused the problem.

Several of the Fortran codes tested contain very long sections of complex coding that tax most compilers. That is, these codes are "compiler breakers." Some compilers have acquired reputations as solid and reliable, while others are recognized as "flaky" and require close watching. Just so, some compilers we used failed to execute the test codes correctly. In some cases, use of single precision caused inaccurate results, but this was not treated as a compiler failure. Generally, the IBM "xlf" Fortran-77 compiler used on the RS 6000 performed extremely well, successfully running every code we tested. Only one irregularity was encountered, in which xlf failed to correctly type a REAL*8 FUNCTION when the REAL*8 typing was included as a prefix to the FUNCTION statement introducing the function definition. We easily worked around this problem by using an untyped FUNCTION statement followed by a REAL*8 type statement typing the function name. This was not truly a bug in the compiler, because the VS Fortran compiler for the 3090 responds the same way, as documented in the VS Fortran manual. Unfortunately, other compilers prefer the REAL*8 as a prefix to the FUNCTION

statement and reject the IBM usage (where "REAL" is the prefix and "*8" is a suffix to the function name). Thus, the only usage accepted by both IBM and non-IBM compilers is a separate type statement to declare the function type.

Some attention was devoted to exercising the PCSIM simulator with a number of PC-DOS software products. Several word processors and text editors were tested, plus a number of graphics programs. The simulator ran all tests perfectly. However, the simulator does not support code invoking a math coprocessor, which could be a significant limitation for some spreadsheets and other computational applications. The Landmark [3] benchmark for PC-DOS systems was run on the simulator, and rated the simulator, running on an RS 6000/320, as half as fast as a 6-MHz PC-AT, or as equally fast as a "turbo" PC-XT with an 8088 running at 8 MHz. Although several simulation parameters could be tuned, the speed of the simulation varied little. The same benchmark ran twice as fast on an RS 6000/530 system.

3. Speed Comparisons

Four classes of speed comparisons were made, including one C-language comparison for speed of integer operations and three Fortran-77 comparisons for speed of floating-point operations. The C-language comparison is the well-known Dhrystone [2] benchmark. The three Fortran-77 comparisons fall into small-, medium-, and large-size codes. The small codes can be run on small personal computers. The medium codes generally cannot be run on small personal computers, but often can be run on systems with several megabytes (Mbytes) of memory. The large codes require more than 20 to 25 Mbytes of memory and 20 to 400 Mbytes of disk space (for output).

Systems available to us for the comparisons were as follows:

- an IBM PC-AT personal computer with an 8-MHz 80286 processor, a 10-MHz 80287 math coprocessor, 640 kbytes of random access memory (RAM), and 70 Mbytes of disk storage;
- a Compaq Deskpro-386 personal computer with a 20-MHz 80386 processor, a 20-MHz 80387 math coprocessor, a 20-MHz Microway Weitek 1167 math coprocessor, 13 Mbytes of RAM, and 300 Mbytes of disk storage;
- a locally integrated personal computer with American Megatrends, Inc. (AMI) motherboard supporting a 20-MHz 80386 processor, a 20-MHz 80387 math coprocessor, 8 Mbytes of RAM, and 140 Mbytes of disk storage;
- a Cray X-MP, available only for the small Fortran-77 benchmarks, with a 7.5-ns cycle time;

- an IBM 3090-300 mainframe with Vector Facility, 64 Mbytes of main memory, 128 Mbytes of extended memory, and over 40 Gbytes of disk storage;
- an IBM RS 6000/320 workstation with a 50-ns clock, 24 Mbytes of RAM, and about 600 Mbytes of disk storage;
- an IBM RS 6000/530 workstation with a 40-ns clock, 48 Mbytes of RAM, and about 1.5 Gbytes of disk storage.

For the tests on the 80286 and 80386 systems, we selected three Fortran-77 compilers:

- an 8-bit compiler, IBM Personal Computer Fortran 2.00, generating library calls for 8087 coprocessor instructions;
- a 16-bit compiler, Ryan-McFarland Fortran 2.11, generating inline code for 80287 instructions;
- a 32-bit compiler, NDP Fortran-386 1.4e supported by Pharlap 386 v2.2 DOS Extender, generating inline code for either the 80387 or Weitek 1167 math coprocessor instructions.

No optimization flags were needed for the IBM compiler or the Ryan-McFarland compiler, which automatically perform available optimizations. The NDP compiler was run with a lower optimization level, "-n3", and a higher optimization level, "-n2 -n3 -OLM", for tests run with the 80387 math coprocessor. Also, the NDP compiler was run with a lower optimization level, "-n4", and a higher optimization level, "-n4 -OLM", for tests run with the Microway Weitek 1167 math coprocessor.

The Cray advanced Fortran-77 compiler "cft77" was used without flags on the Cray X-MP, since it performs full optimization by default.

For the IBM 3090, IBM VS Fortran Release 2 was used at optimization level 3 without vectorization (NOVECTOR), and at optimization level 3 with vectorization (VECTOR). Few of the codes tested could benefit from vectorization. All that could so benefit were tested with the vectorization option.

The RS 6000 native Fortran-77 compiler xlf was used with no optimization and with optimization level "-QOPT" for the tests. Since none of the Fortran codes tested could benefit from the inlining option, we have not included any results with the inlining option. ("Inlining" is replacing a subprogram CALL with an inline expansion of the body of the subprogram called, eliminating the

overhead of the CALL and allowing the subprogram to be incorporated into optimization done by the compiler for the calling program.)

Our Dhrystone [2] benchmark has been ported from the Ada language to C. It is intended to resemble "typical" programs, and does not use floating-point operations. Obviously, it is not typical of scientific or engineering programs, but characterizes data-handling problems. We used Borland Turbo C Version 2.0 to compile the PC-DOS tests, Cray "cc" to compile the Cray X-MP test, and RS 6000 "xlc" to compile the RS 6000 tests. The latter was used (1) without optimization and (2) with optimization and inlining "-O -Q" in the tests. The PC-DOS version of the benchmark was also run under the PCSIM emulator on the RS 6000 systems.

3.1 Dhrystone Comparison

The Dhrystone test results are shown in table 1. No test was run on the IBM 3090 because a C compiler was not available on that system. Surprisingly, the RS 6000 systems run the benchmark 2.5 to 3.0 times faster than the Cray X-MP. The Cray architecture, which is optimized for floating-point operations rather than byte operations, takes about 17 times more clock cycles to perform the benchmark than the RS 6000. The PCSIM result is comparable to the Dhrystone speed of a PC-XT running a 4.77-MHz 8088. For comparison, a DEC VAX 11/780 runs close to 1600 Dhrystones/s.

Table 1. Dhrystone benchmark results

[Best optimizations were used in each case except for the RS 6000 results, which include (1) no optimization, (2) -O standard optimization, and (3) -O -Q standard optimization and inlining (inline expansion of short subprogram units, avoiding overhead of a CALL.]

Configuration	Compiler	No. Loops	Time (s)	Dhrystones/s
RS 6000/530	xlc -O -Q	500,000	6.17	81,037
RS 6000/320	xlc -O -Q	500,000	7.69	65,020
RS 6000/530	xlc -O	500,000	8.03	62,267
RS 6000/320	xlc -O	500,000	9.76	51,230
Cray X-MP (one CPU)	Cray cc	500,000	18.564	26,934
RS 6000/320	xlc (no opt)	500,000	20.81	24,027
20-MHz 80386/80387[a]	Turbo C	50,000	7.91	6,321
8-MHz 80286/80287[b]	Turbo C	50,000	29.0	1,724
RS 6000/320	PCSIM[c]	50,000	145	344

[a] *64-kbyte memory cache used.*

[b] *10-MHz 80287, but Dhrystone does not use floating-point arithmetic.*

[c] *PCSIM is a software emulation of an 8088 running PC-DOS.*

3.2 Whetstone Comparison

The Whetstone [4] comparison is widely used to estimate the floating-point efficiency of processor/compiler systems. It incorporates generally non-vectorizable loops to perform basic arithmetic and transcendental function evaluations. A significant fraction of the benchmark is devoted to overhead of subprogram CALLs and loop iteration. The Cray X-MP does not implement REAL*4 precision and appears only in the double-precision small code comparisons. Single-precision Whetstones are given in table 2 and double-

Table 2. Results of single-precision Whetstone benchmark

[Full optimizations were used where compiler flags are not indicated; note: SP = single precision.]

Configuration	Compiler	No. SP Mega Whetstones	Time (s)	SP Mega Whetstones/s
RS 6000/530	xlf -QOPT	100	3.96	25.3
IBM 3090/VF	IBM VS Fortran Rel. 2, OPT(3)	1000	42.94	23.3
IBM 3090/VF	IBM VS Fortran Rel. 2, OPT(3) VECTOR	1000	43.70	22.9
RS 6000/320	xlf -QOPT	100	4.93	20.3
RS 6000/530	xlf (no opt)	100	7.16	14.0
RS 6000/320	xlf (no opt)	100	8.84	11.3
20-MHz 80386/*mW*1167[a]	NDP Fortran-386 1.4e, Pharlap 386 v2.2, -n4 -OLM	100	27.95	3.58
20-MHz 80386/*mW*1167[a]	NDP Fortran-386 1.4e, Pharlap 386 v2.2, -n4	100	29.16	3.43
20-MHz 80386/80387[b]	NDP Fortran-386 1.4e, Pharlap 386 v2.2, -n2 -n3 -OLM	100	51	1.96
20-MHz 80386/80387[b]	NDP Fortran-386 1.4e, Pharlap 386 v2.2, -n3	100	63	1.59
20-MHz 80386/80387[b]	RM Fortran 2.11	10	10.10	0.990
20-MHz 80386/80387[b]	IBM PC Fortran 2.00/8087	10	37.85	0.264
8-MHz 80286/80287[c]	RM Fortran 2.11	10	49.17	0.203
8-MHz 80286/80287[c]	IBM PC Fortran 2.00/8087	10	133.80	0.075

[a] *Microway Weitek 1167 coprocessor used.*

[b] *64-kbyte memory cache used.*

[c] *10-MHz 80287 used.*

precision Whetstones are given in table 3. RS 6000 models 320 and 530 tightly bracket the performance of the mainframe IBM 3090 processor. Use of the Vector Facility gives no improvement because of the absence of vectorizable loops in the code. The RS 6000 systems run about one order of magnitude faster than well-equipped 80386 systems and over two orders of magnitude faster than older 80286 systems. Significantly, the model 530 runs the double-precision benchmark as fast as the Cray X-MP.

Table 3. Results of double-precision Whetstone benchmark

[Full optimizations were used where compiler flags are not indicated; note: DP = double precision.]

Configuration	Compiler	No. DP Mega Whetstones	Time (s)	DP Mega Whetstones/s
RS 6000/530	xlf -QOPT	100	3.61	27.7
Cray X-MP (one CPU)	Cray cft77	100	3.641	27.5
RS 6000/320	xlf -QOPT	100	4.45	22.5
IBM 3090/VF	IBM VS Fortran Rel. 2, OPT(3)	1000	50.68	19.7
IBM 3090/VF	IBM VS Fortran Rel. 2, OPT(3) VECTOR	1000	51.27	19.5
RS 6000/530	xlf (no opt)	100	6.88	14.5
RS 6000/320	xlf (no opt)	100	8.56	11.7
20-MHz 80386/*mW*1167[a]	NDP Fortran-386 1.4e, Pharlap 386 v2.2, -n4 -OLM	100	34.05	2.94
20-MHz 80386/*mW*1167[a]	NDP Fortran-386 1.4e, Pharlap 386 v2.2, -n4	100	34.49	2.90
20-MHz 80386/80387[b]	NDP Fortran-386 1.4e, Pharlap 386 v2.2, -n2 -n3 -OLM	100	54	1.85
20-MHz 80386/80387[b]	NDP Fortran-386 1.4e, Pharlap 386 v2.2, -n3	100	66	1.52
20-MHz 80386/80387[b]	RM Fortran 2.11	10	11.21	0.892
8-MHz 80286/80287[c]	RM Fortran 2.11	10	54.87	0.182
20-MHz 80386/80387[b]	IBM PC Fortran 2.00/8087	10	75.46	0.133
8-MHz 80286/80287[c]	IBM PC Fortran 2.00/8087	10	258	0.039

[a] *Microway Weitek 1167 coprocessor used.*
[b] *64-kbyte memory cache used.*
[c] *10-MHz 80287 used.*

3.3 F4T Comparison

The F4T code is a Fortran-coded fast Fourier transform, written by one of the authors (Wyatt), which modifies an in-place decimation in time in a way that eliminates the need for a separate bit-reversal sort of the data order. The algorithm uses a rapidly varying stride to access and store the data. The algorithm defeats most vectorizing compilers, although the code can, in principle, be vectorized. Optimization of the inner loop of the code is a strong test of a compiler's optimization skills, because the inner loop contains two independent indices, 14 index computations, and 20 floating-point computations. A clever compiler has many opportunities to reorder code. The transform size done in this benchmark is 1024 points, using 2048 data words. Single-precision results are shown in table 4 and double-precision results are shown in table 5.

Table 4. Results of single-precision F4T benchmark

[Full optimizations were used where compiler flags are not indicated; note: SP = single precision.]

Configuration	Compiler	No. SP F4T Loops	Time (s)	SP F4T Loops/s
RS 6000/530	xlf -QOPT	1000	7.44	134.4
RS 6000/320	xlf -QOPT	1000	9.35	107.0
IBM 3090/VF	IBM VS Fortran Rel. 2, OPT(3) VECTOR	1000	9.37	106.7
IBM 3090/VF	IBM VS Fortran Rel. 2, OPT(3)	1000	9.40	106.4
RS 6000/530	xlf (no opt)	1000	31.52	31.7
RS 6000/320	xlf (no opt)	1000	38.66	25.9
20-MHz 80386/*mW*1167[a]	NDP Fortran-386 1.4e, Pharlap 386 v2.2, -n4 -OLM	1000	106.88	9.36
20-MHz 80386/*mW*1167[a]	NDP Fortran-386 1.4e, Pharlap 386 v2.2, -n4	1000	120.39	8.31
20-MHz 80386/80387[b]	NDP Fortran-386 1.4e, Pharlap 386 v2.2, -n2 -n3 -OLM	100	26.19	3.82
20-MHz 80386/80387[b]	NDP Fortran-386 1.4e, Pharlap 386 v2.2, -n3	100	27.74	3.60
20-MHz 80386/80387[b]	RM Fortran 2.11	100	40.81	2.45
20-MHz 80386/80387[b]	IBM PC Fortran 2.00/8087	100	170.93	0.585
8-MHz 80286/80287[c]	RM Fortran 2.11	10	22.95	0.436
8-MHz 80286/80287[c]	IBM PC Fortran 2.00/8087	100	608.19	0.164

[a]*Microway Weitek 1167 coprocessor used.*

[b]*64-kbyte memory cache used.*

[c]*10-MHz 80287 used.*

A detailed count of operations shows that, for a 1024-point transform, F4T executes a total of 55,322 floating-point operations and 21 transcendental function evaluations. Further, 38,642 integer operations are possible, although compiler optimization may reduce this number substantially. Additional operations related to DO-loop index updating and branching are performed for 6,237 loops. Using the elapsed times tabulated for double-precision F4T, we calculate the following floating-point speeds in Mflops (million floating-point operations per second):

Cray X-MP	10.0
Model 530	9.0
Model 320	7.2
3090/VF	5.9
PC equipped with Weitek 1167	3.6
PC equipped with 80387	2.0

These figures are consistent with most published data, we believe. For example, IBM advertises double-precision speeds of 7.4 and 10.9 Mflops for

Table 5. Results of double-precision F4T benchmark

[Full optimizations were used where compiler flags are not indicated. Entries marked "fail" signify that the code failed to run properly because of bad code generated by the compiler; note: DP = double precision.]

Configuration	Compiler	No. DP F4T Loops	Time (s)	DP F4T Loops/s
Cray X-MP (one CPU)	Cray cft77	1000	5.55	180.2
RS 6000/530	xlf -QOPT	1000	6.18	161.8
RS 6000/320	xlf -QOPT	1000	7.68	130.2
IBM 3090/VF	IBM VS Fortran Rel. 2, OPT(3) VECTOR	1000	9.36	106.8
IBM 3090/VF	IBM VS Fortran Rel. 2, OPT(3)	1000	9.45	105.8
RS 6000/530	xlf (no opt)	1000	30.58	32.7
RS 6000/320	xlf (no opt)	1000	38.03	26.3
20-MHz 80386/*mW*1167[a]	NDP Fortran-386 1.4e, Pharlap 386 v2.2, -n4 -OLM	1000	154.89	6.46
20-MHz 80386/80387[b]	NDP Fortran-386 1.4e, Pharlap 386 v2.2, -n2 -n3 -OLM	1000	277.05	3.61
20-MHz 80386/80387[b]	RM Fortran 2.11	100	49.10	2.04
20-MHz 80386/80387[b]	IBM PC Fortran 2.00/8087	100	271.34	0.369
8-MHz 80286/80287[c]	IBM PC Fortran 2.00/8087	10	93.75	0.107
8-MHz 80286/80287[c]	RM Fortran 2.11	10	fail	fail

[a] *Microway Weitek 1167 coprocessor used.*

[b] *64-kbyte memory cache used.*

[c] *10-MHz 80287 used.*

the model 320 and model 530, respectively. The low figure for the Cray X–MP is not surprising for an unvectorized code. Although single-processor speeds up to 200 Mflops are possible for highly vectorized code on the X–MP, a 10-Mflop rating is representative of its scalar processing speed [5].

The optimization power of the RS 6000 xlf compiler is displayed by the factor-of-five speedup in run times for optimized code over unoptimized code. This strongly demonstrates the effectiveness of the optimizations done by this compiler. At single precision, the 320 matches the 3090, and the 530 runs the comparison about 25 percent faster. At double precision, the 320 runs about 25 percent faster and the 530 runs about 50 percent faster than the 3090, at top optimization. The Cray X-MP manages to squeak out a 10-percent faster performance than the RS 6000/530. The RS 6000 always runs our double-precision benchmarks faster than our single-precision benchmarks. Compared to the Whetstone benchmark, the 80386 and 80286 systems perform the F4T benchmark relatively slower, running from 15 to 1500 times slower than the RS 6000/530.

3.4 SVD Comparison

The last small code used is the singular value decomposition (SVD) driver and supporting routines taken from LINPACK [6], with an outer loop which we added. To avoid various error terminations, we removed the overflow test portion of the driver. Comparisons were not run on the IBM 3090 or the Cray X-MP. Generally, timing comparisons are similar to those for the Whetstone benchmark. Single-precision results are given in table 6 and double-precision results are given in table 7.

Table 6. Results of single-precision SVD benchmark

[Full optimizations were used where compiler flags are not indicated; note: SP = single precision.]

Configuration	Compiler	No. SP SVD Loops	Time (s)	SP SVD Loops/s
RS 6000/530	xlf -QOPT	100	7.00	14.3
RS 6000/320	xlf -QOPT	100	8.68	11.5
RS 6000/530	xlf (no opt)	100	30.25	3.31
RS 6000/320	xlf (no opt)	100	37.64	2.66
20-MHz 80386/*mW*1167[a]	NDP Fortran-386 1.4c, Pharlap 386 v2.2, -n4 -OLM	100	78	1.28
20-MHz 80386/80387[b]	NDP Fortran-386 1.4c, Pharlap 386 v2.2, -n2 -n3 -OLM	1000	457	0.219
20-MHz 80386/80387[b]	RM Fortran 2.11	100	516	0.194

[a] *Microway Weitek 1167 coprocessor used.*

[b] *64-kbyte memory cache used.*

Table 7. Results of double-precision SVD benchmark
[Full optimizations were used where compiler flags are not indicated; note: DP = double precision.]

Configuration	Compiler	No. DP SVD Loops	Time (s)	DP SVD Loops/s
RS 6000/530	xlf -QOPT	100	6.86	14.6
RS 6000/320	xlf -QOPT	100	8.62	11.6
RS 6000/530	xlf (no opt)	100	32.25	3.10
RS 6000/320	xlf (no opt)	100	40.03	2.50
20-MHz 80386/*mW*1167[a]	NDP Fortran-386 1.4e, Pharlap 386 v2.2, -n4 -OLM	100	144	0.694
20-MHz 80386/80387[b]	NDP Fortran-386 1.4e, Pharlap 386 v2.2, -n2 -n3 -OLM	1000	450	0.222
20-MHz 80386/80387[b]	RM Fortran 2.11	100	523	0.191

[a]*Microway Weitek 1167 coprocessor used.*
[b]*64-kbyte memory cache used.*

3.5 Comparison of Medium-Size Codes

A number of well-exercised computer codes are maintained at our facility for the purpose of modeling the physics of EMP generation by nuclear bursts. Three medium-size codes were tested which manipulate large amounts of data that cannot fit in the data cache of RS 6000 systems. Thus, the RS 6000 systems would be forced to move large amounts of data to and from main memory, with fewer cache hits. Although runnable on fast 80386 systems with sufficient RAM and disk space, these codes are too large to run on small personal computers. One code, CHAP, was vectorized on the IBM 3090 but would fail most of the time because of bad code generated by the compiler in one section of the code. Another code, LEMP2, is known to vectorize well on the Cray X-MP, but failed to vectorize significantly on the IBM 3090. Both single- and double-precision comparisons are given in table 8. Unfortunately, the Cray X-MP was not available for comparison runs of the medium and large codes.

Clearly, only the most advanced compilers could handle these codes. Although all compiled "successfully," none of the 80386 compilers ran any of the three codes successfully or obtained the right answers in a reasonable amount of time, except the Weitek 1167 compilation by NDP Fortran. Oddly, the CHAP runs failed at both single and double precision with vectorization attempted on the IBM 3090 system, although the double-precision version would run properly occasionally when optimization was defeated in one loop. CPU time for one successful run has been entered in the table in parentheses. The RS 6000/530 times are close to the unvectorized 3090 times, with the model 320 trailing slightly farther back. Relatively little optimization is being done by the compiler to these codes, probably because of the very long main

Table 8. Results of medium-size-code benchmark

[All times are in seconds. Entries marked "N/A" signify that the code was too large to run in 640 kbytes of memory, the maximum available. Entries marked "fail" signify that the code failed to run properly because of bad code generated by the compiler or associated assemblers and linkers. Dashes indicate that the run was not attempted. Note: SP = single precision; DP = double precision.]

System	CHAP (SP)	CHAP (DP)	LEMP2 (SP)	LEMP2 (DP)	GLANCL (SP)	GLANCL (DP)
IBM 3090/VF with VS Fortran VECTOR	fail	(123.42)[a]	—	2638.22	—	39.48
IBM 3090/VF with VS Fortran NOVECTOR	243.05	319.61	—	—	—	39.82
RS 6000/530 with xlf -QOPT	329.92	295.19	5314.88	2510.12	45.37	42.28
RS 6000/320 with xlf -QOPT	416.05	384.06	6691.63	3238.26	58.50	54.07
RS 6000/530 with xlf (no opt)	568.46	567.51	—	—	74.00	75.30
RS 6000/320 with xlf (no opt)	716.38	733.57	—	—	94.69	95.60
80386/*mW*1167 & NDP Fortran	3134	fail	—	35,782	—	—
80386/80387 & NDP Fortran	fail	fail	—	fail	—	—
80386/80387 & RM Fortran	9035[b]	N/A	N/A	N/A	—	—

[a] *Usually failed.*

[b] *Ran to completion but exponent range too small for correct results.*

programs (about 1500 to 2000 lines of non-comment code), and because of the complexity of the loops being used. All three codes were written in a "1960's" programming style with little modularity and a great deal of store-and-fetch of temporary results.

3.6 Comparison of Large-Size Codes

Finally, we present comparisons for three codes with large memory and large computation requirements typical of recent developments in computer modeling of radiation transport (LHAPHEN) and solution of Maxwell's equations (ROUNDSY and ROUND3Y). Results for the working precision of each code are given in table 9.

The first code, LHAPHEN, is a nonvectorizable code for Monte Carlo simulation of neutron and gamma radiation transport. It is composed of many subprogram units, each performing some function for the overall simulation.

Table 9. Results of large-code benchmark

[All times are in seconds. Dashes indicate that the run was not attempted. Vectorized code was attempted only for ROUND3Y on the IBM 3090, because the other two codes contained no significant vectorizable computations. Available memory was too small on the RS 6000/320 to run ROUNDSY or ROUND3Y.]

System	LHAPHEN	ROUNDSY	ROUND3Y
IBM 3090/VF with VS Fortran VECTOR	—	—	6309.69
IBM 3090/VF with VS Fortran NOVECTOR	2712.45	4151.85	13,500.63
RS 6000/530 with xlf -QOPT	2386.57	5905.96	14,507.51
RS 6000/320 with xlf -QOPT	3117.96	—	—

Array storage is extensive, but within the range of a 20-Mbyte memory system. Because it processes one particle at a time, code is executed in straight-line fashion through dozens of subprograms, repeated in a large outer loop in the main program. Although an ideal candidate for some massively parallel computers where each subprocessor tracks one particle, the code does not vectorize below the one-particle level. For this code, performance by the RS 6000 systems brackets the mainframe IBM 3090 performance.

The second code, ROUNDSY, performs complicated interpolation of tri-cubic spline hypersurfaces, with multiple nested short DO-loops. It generates data in very large arrays, on the order of 30 Mbytes in size, and stores the data on disk. In its original form this code used REAL*16 precision for some of the interpolation, but was converted to use REAL*8 operands, instead, for the purposes of this study. Because of the way the interpolation is done, negligible vectorization can be done in this code as it is written. The IBM 3090 runs this code in about 70 percent of the time used by the RS 6000/530. This difference is probably due to a very low cache hit rate on the 530 and its relatively slower memory system, compared with the 3090's fast 64-Mbyte main memory.

The third code, ROUND3Y, solves a three-dimensional finite-difference model of Maxwell's equations using a very large mesh and very large arrays. The code was written to be easily vectorizable. The dimensionality of the mesh arrays is $220 \times 151 \times 16$, and all computations are done in REAL*8 precision. For this comparison, the dimensionality was reduced, ultimately, to $220 \times 151 \times 4$ to fit in the 48-Mbyte memory of the RS 6000/530 system used. (A maximum of 128 Mbytes of RAM can be installed in the model 530.) The full mesh ($220 \times 151 \times 16$) requires a 95-Mbyte region to run on the IBM 3090, and processing slows by one-third because of intense paging between the 64-Mbyte main memory and expanded memory. A workset of 40 Mbytes is mandatory for the full-sized mesh; otherwise, thrashing sets in on the 3090. When a reduced mesh of $220 \times 151 \times 8$ was attempted on the RS 6000/530, which should "barely fit," the system crashed when array-filling caused the workset to reach 40 Mbytes (on the virtual memory system of the 530). Hence,

the final reduced mesh of 220 × 151 × 4 was attempted, with a successful outcome. When the final mesh was used, we observed a workset of 28 Mbytes during running on the 530. Without vectorization, runtimes were comparable for the 530 and the 3090; but with vectorization, the 3090 gained slightly over a factor of two in speed.

4. Conclusions

Our comparisons clearly place the RS 6000 workstations close to mainframe computers in speed and memory capacity. Although vectorizable (or possibly parallelizable) codes can run much faster on some vector (or parallel) machines, the RS 6000 design runs scalar, floating-point-intensive codes at mainframe or even supercomputer speed. For integer and byte processing typified by the Dhrystone benchmark, the RS 6000 systems turn in very high numbers, running up to 81,037 Dhrystones/s, three times faster than a Cray X-MP (single processor).

The Fortran-77 compiler xlf, native to the RS 6000, successfully compiled and ran all our test programs, with and without optimization. We used several of our favorite "compiler breakers" without any problems occurring. Optimizations performed by this compiler are of very high quality, as shown by our F4T test, leading to speedups of a factor of five over unoptimized code.

The PC-DOS emulator software appears quite robust, running even direct-screen writers and VGA graphics with no difficulty. Of course, performance is slow for this software emulator, and was seen to be close to that of 4.77-MHz 8088 or 6-MHz 80286 systems.

Performance by the RS systems was somewhat limited by slowness of the main memory system, compared to the speed of the memory cache. Small codes with small amounts of data were well supported by the 64-kbyte data cache and 8-kbyte instruction cache of the RS systems, and ran fastest. Medium and large codes tended to run somewhat slower on the RS systems, relative to the IBM 3090 which has a faster main memory. In the case of medium and large codes that use larger amounts of data than can fit in the data cache, the fast multiply-and-add operation on the RS systems may be "data-starved" because of the relatively longer time needed to fetch operands from memory. In the case of small loops using cached data only, tests show that the 3090 Vector Facility can execute a (vector) multiply-and-add every three clock periods of 25 ns each (issuing one multiply every two periods, and one add every period), for a rate of 26.7 Mflops. For the same small loops using only cached data, the RS 6000/530 can, theoretically, execute a (scalar) multiply-and-add every clock period of 40 ns (with a two-period latency), for a rate of 50 Mflops. However, we did not achieve this scalar rate in any of our benchmarks.

All small codes and the Dhrystone benchmark ran almost exactly 25 percent faster on the model 530 than on the model 320, because of the 25-percent-faster clock speed on the 530. Because the 32-kbyte cache and 64-bit-wide memory bus on the model 320 are adequate for these small codes, only the faster clock of the model 530 causes faster execution. Similarly, single-precision versions of CHAP and LEMP2 show this same characteristic. However, double-precision versions of CHAP and LEMP2, and the standard version of LHAPHEN, show a 3- to 4-percent improvement in speed on the 530 over and above the effect of the faster clock. This implies that there is potential "data starvation" in the model 320 for these codes, which is ameliorated by the larger 64-kbyte data cache and 128-bit-wide memory bus of the model 530. It appears that ROUNDSY execution is slowed by nearly a factor of two because of data starvation. As with virtually all codes and computers, a careful rewrite of the inner loops of these codes may help alleviate the slowdown by making better use of the data cache.

The "super-scalar" ability of the RS 6000 CPU to issue and execute up to four instructions in parallel in each clock period allows simple add, multiply, or multiply-and-add loops to be done with no loop overhead, as has been mentioned previously. Thus the loops are executed at "vector speed," with one result operand produced each clock period. The pipeline latency is one or two clock periods for cached data. Pipeline latency increases by memory access time for data not in cache, and we have seen that memory bandwidth (400 Mbytes/s for a 40-ns clock) can supply only 50 million 8-byte operands per second. A significant enhancement to the RS 6000 design, providing a capability equivalent to vector performance on simple loops, would involve doubling the memory bandwidth and providing mechanisms to start data "fetch-ahead" so that the CPU does not become data-starved. Such a feature might increase processing speeds by a factor of two for many vectorizable codes using unit stride, as indicated by our tests with ROUND3Y.

In the past, we have tried to upgrade personal computers in speed, memory, and storage capacity to accommodate the running of various mainframe computer codes, generally falling short of a satisfactory result. Even upgrades to an 80486 processor would fall short of RS 6000 performance by roughly a factor of four to five. Given the superior integration of CPU, memory, storage, and software of the RS 6000, and the observed and potential speed advantage of the RS 6000, augmenting personal computers will remain an inferior strategy.

In conclusion, it is clear that the RS 6000 models tested are powerful enough to allow routine running of EMP codes and the like in times comparable to mainframe supercomputer execution times. The proximity of the user and his interactive graphical interface to the running EMP code could stimulate a "quantum" leap in visualization of the physics being modeled and insight into the EMP phenomenon.

5. References

[1] *IBM RISC System/6000 Technology*, edited by Mamata Misra, International Business Machines Publication SA23-2619 (1990).

[2] Reinhold P. Weicker, *Dhrystone Benchmark Program Version C/ 1.1*, CACM **27**(10) (October 1984), p 1013; translated from Ada by Rick Richardson, 1 June 1986.

[3] *Landmark CPU Speed Test: Speed Version 1.05*, Landmark Software, 1142 Pomegranate Court, Sunnyvale, CA 94087 (1986).

[4] H. J. Curnow and B. A. Wichmann, *A Synthetic Benchmark*, Computer J. **19** (1) (February 1976), pp 43–49.

[5] J. M. Levesque and J. W. Williamson, *A Guidebook to Fortran on Supercomputers*, Academic Press, Inc., San Diego, CA (1989); see figure 4.35 for an example for an 8.5-ns clock X-MP.

[6] J. J. Dongarra, C. Moler, J. Bunch, and G. W. Stewart, *Linear Equation Package (LINPACK)*, Applied Mathematics Division, Argonne National Laboratory (19 March 1979).

DISTRIBUTION

Administrator
Defense Technical Information Center
Attn DTIC-DDA (2 copies)
Cameron Station, Building 5
Alexandria, VA 22304-6145

Director
Defense Nuclear Agency
Attn RAAE, Atmospheric Effects Div
Attn RAEE, EMP Effects Div
Attn TISI, Scientific Information Div
Attn RAEV, Electronics Vulnerability Div
Washington, DC 20305

Under Secretary of Defense Research
& Engineering
Attn Technical Library, 3C128
Washington, DC 20301

Director
US Army Ballistics Research Laboratory
Attn SLCBR-DD-T (STINFO)
Aberdeen Proving Ground, MD 21005

US Army Electronics Technology &
Devices Laboratory
Attn SLCET-DD
Ft. Monmouth, NJ 07703

Commanding Officer
US Army Foreign Science & Technology Center
Attn AMXST-SC, Sciences Div
220 Seventh Street, NE
Charlottesville, VA 22901

Director
US Army Materiel Systems Analysis Activity
Attn AMXSY-MP
Aberdeen Proving Ground, MD 21005

Director
US Army Missile Laboratory (USAMICOM)
Attn AMSMI-RPT, Technical Information Div
Redstone Arsenal, AL 35809

Commander
US Army Nuclear & Chemical Agency
Attn Dr. Adam Renner, MONA-NU
7500 Backlick Road, Building 2073
Springfield, VA 22150

US Chief of Army Research Office
Attn SLCRO-MA, Dir Mathematics Div
Attn SLCRO-PH, Dir Physics Div
PO Box 12211
Research Triangle Park, NC 27709-2211

Director
Naval Research Laboratory
Attn 2600, Technical Info Div
Washington, DC 20375

Commander
Naval Surface Weapons Center
Attn E-43, Technical Library
White Oak, MD 20910

International Business Machines Corporation
Attn R. Cimini
6705 Rockledge Drive
Bethesda, MD 20817

Lawrence Livermore Laboratory
Attn Technical Library
PO Box 969
Livermore, CA 94550

Director
National Institute of Standards & Technology
Attn Library
Washington, DC 20234

DISTRIBUTION

Sandia National Laboratories
Attn Technical Library
PO Box 5800
Albuquerque, NM 87185

US Army Laboratory Command
Attn Technical Director, AMSLC-TD

Installation Support Activity
Attn Legal Office, SLCIS-CC

USAISC
Attn Admin Ser Br, AMSLC-IM-VA
Attn Tech Pub Br, AMSLC-IM-VP

Harry Diamond Laboratories
Attn Laboratory Directors
Attn Library, SLCHD-TL (3 copies)
Attn Library, SLCHD-TL (WRF)

Harry Diamond Laboratories (cont'd)
Attn Chief Scientist, SLCHD-CS
Attn Chief, SLCHD-NW
Attn Chief, SLCHD-NW-CS
Attn Chief, SLCHD-NW-E
Attn Chief, SLCHD-NW-EH
Attn Chief, SLCHD-NW-EP
Attn Chief, SLCHD-NW-ES
Attn Chief, SLCHD-NW-P
Attn Chief, SLCHD-NW-R
Attn Chief, SLCHD-NW-RP
Attn Chief, SLCHD-NW-RS
Attn Chief, SLCHD-NW-TN
Attn Chief, SLCHD-NW-TS
Attn G. Merkel, SLCHD-NW-TN
Attn M. Bushell, SLCHD-NW-RS
Attn R.J. Chase, SLCHD-NW-EP
Attn C. S. Kenyon, SLCHD-NW-EP
Attn W.T. Wyatt, SLCHD-NW-EP (30 copies)